Data-Driven Characterization and Modeling of Web Map System Workload

Vinicius Vinicius Gonçalves Braga, Sand Luz Correa, Kleber Vieira Cardoso, Aline Carneiro Viana

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 31, 2021, accepted February 8, 2021. Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2021.3058622

Data-Driven Characterization and Modeling of Web Map System Workload

VINÍCIUS GONÇALVES BRAGA[1], SAND LUZ CORREA[1], KLEBER VIEIRA CARDOSO[1], AND ALINE CARNEIRO VIANA[2]

[1]Institute of Informatics (INF), Universidade Federal de Goiás (UFG), Goiânia 74690-900, Brazil
[2]Inria, 91120 Palaiseau, France

Corresponding author: Sand Luz Correa (sandluz@ufg.br)

This work was supported in part by the project EMBRACE Inria Associated-Team, and in part by the Fundação de Amparo à Pesquisa do Estado de Goiás (FAPEG) in the call No. 05/2016, under Grant 201610267001193 and also in the call No. 03/2015 FAPEG/CNPq/PPP, Grant 201810267001734.

ABSTRACT Every month, billions of users access Web Map Systems (WMSs), such as Google Maps, to visualize geospatial data. A large number of users and the huge amount of data demanded by these applications make the design and development of WMSs a challenging task, especially in terms of performance and scalability. In this context, workload generators become crucial tools, as they help system administrators to plan the capacity of WMSs and design provisioning strategies for peak load scenarios. However, little is known about the workload patterns generated by WMS users. In this work, we use data anonymously collected from sessions of a client application of Google Maps to devise a model that describes how users of desktop terminals navigate in a Web map. Based on this model, we implement a workload generator called MUSeGen. We compare the workload patterns generated by MUSeGen against the workload patterns found in real data. Results show that MUSeGen generates synthetic traces whose navigation patterns closely match those found in real data. We also compare MUSeGen against HELP, a workload generator built upon previous findings on empirical knowledge on the usage of WMSs. Results show that the number of issued operations per session in HELP is, on average, four times lower than that in MUSeGen and the number of tiles requested is, on average, twice lower than that in our tool. In addition, navigation patterns in HELP are much simpler than in MUSeGen. These findings support the conclusion that MUSeGen produces more realistic workloads than HELP. To illustrate how such differences affect performance evaluation in practice, we carry out a performance evaluation of a real WMS under workloads generated by HELP and MUSeGen. Our evaluation shows that the system capacity under HELP is three times less than that obtained under MUSeGen, highlighting the value of MUSeGen.

INDEX TERMS Web map user, tile-based systems, workload model, workload generation, performance evaluation.

I. INTRODUCTION

Geographic Information Systems (GIS) provide access to information associated with a specific position on a map. Usually, this is done by offering the overlay of different georeferenced data layers, which opens an opportunity to explore such data in many types of analysis and investigation. A common use of this technology is for the creation of Web Map Systems (WMSs) such as Google Maps, Bing Maps, and OpenStreetMap.

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Marozzo.

Every month, billions of users access WMSs to visualize maps and obtain geographic orientation [1]. Currently, video, gaming, and maps are applications that together account for 70% of all Internet traffic and this number is expected to grow in the near future [2]. A large number of users and the huge amount of data demanded by GIS applications make the design and development of WMSs a challenging task, especially in terms of performance and scalability.

In this context, workload models and workload generators are crucial tools, as they help system administrators to plan the capacity of WMS and design provisioning strategies for peak load scenarios. These tasks are essential to ensure a better user experience. For the specific case of WMSs,

the workload consists of user requests to GIS resources in a Web server. Such requests are generated in response to operations (mainly pan and zoom) issued by the user during the map navigation session. In this work, we focus on this type of workload.

However, despite the importance and scale of WMSs, there is little knowledge about the workload patterns generated by users of such systems. As far as we know, no previous work has addressed the problem of characterizing WMS user behavior using data collected from real WMS sessions. The lack of publicly available datasets has notably contributed to this context. To illustrate this gap, a survey on workload generators for Web-based systems [3] reviewed more than 20 works on this topic in different types of applications (e.g., online social networks, video stream, and cloud computing services). None of them, however, focus on WMSs. Yet, WMSs are important systems whose user navigation patterns are very different from those encountered in non-geospatial Web systems [4], [5]. For example, while navigation on the latter occurs mostly through hyperlinks, WMS users navigate by zooming and panning on the map.

Indeed, we are aware of only one work, the HELP model [5], that statistically characterizes how users of desktop terminals navigate during a WMS map navigation session. However, HELP is not entirely based on real-world data and lacks detailed knowledge of how WMSs are used in practice.

In this article, we present a characterization of how users of desktop terminals browse a WMS in a client application of Google Maps. This characterization is conducted based on data gathered anonymously from user sessions of Google Maps, using an extension that we developed for the Google Chrome browser [6]. Based on this characterization, we propose a model that statistically describes user actions in a WMS session. Particularly, the proposed model captures how users perform the operations available in the system within a navigation session, how these operations are distributed over time, and the sequence that they are issued. Using this information, our model estimates the resulting workload by translating user actions into GIS requests.

Next, we use the proposed model to drive the development of a workload generator for WMSs called *Maps User Session Workload Generator* (MUSeGen). We illustrate the effectiveness of MUSeGen by comparing the workload patterns generated by our tool against the workload patterns found in real data. Results show that MUSeGen generates synthetic traces whose navigation patterns (e.g., mean number of issued operations per session, mean session length, mean pan sizes, and mean zoom jump sizes) closely match those found in real data. We also compare MUSeGen against HELP. Results show that, on average, the number of issued operations per session in HELP is four times lower than that in MUSeGen, and the number of tiles requested to the server is twice lower than that in our tool. In addition, navigation patterns (pan and zoom) in HELP are much simpler than in MUSeGen. For example, in HELP, the size of the pan movement is fixed and the direction of the movement is limited to a few values. These findings support the conclusion that MUSeGen produces more realistic workloads than HELP. To illustrate how such differences affect the performance evaluation in practice, we carry out a performance evaluation of a real WMS under workloads generated by different tools, namely: (i) a stress testing software; (ii) HELP; and (iii) MUSeGen. Our evaluation confirms the limitation of the stress testing tool in generating different workloads under different conditions. Our evaluation also shows that the system capacity under HELP is three times less than that obtained under MUSeGen. This happens because the number of tiles requested per zoom or pan in MUSeGen is higher than in HELP. Thus, most of the requests in MUSeGen are satisfied by the browser's cache, increasing the system ability to serve more users concurrently.

In summary, the main contributions of this work are the following:

- We detail our data collection methodology for developing a Google Chrome extension that anonymously collects data from a client application of Google Maps. For five months, our data collection campaign gathered information from 146 users (of desktop terminals) living in an area of nearly 7,300 km^2. This campaign resulted in a dataset containing a total of 36,860 URLs.
- Using the collected data, we propose a model that describes how users of desktop terminals browse a WMS. Although access to Google Maps using mobile terminals is a growing trend, desktop terminals are still responsible for nearly 50% of global Google Maps usage [7]. In addition, we believe that the methodology applied in this work can be used to build workload models for mobile terminals. This is so since the aspects captured in our model (e.g., distribution of the interactions over time, pan operation, zoom operation, the sequence of operations, and the number of tiles requested to the server) are also relevant for describing mobile user behavior.
- We develop a workload generator (MUSeGen) that reproduces user actions within WMS sessions and validate the user behavior generated by our tool against patterns observed in real traces as well as in traces generated by HELP.
- We make MUSeGen as well as the empirical data used in its parameterization publicly available to the community.[1]
- We assess the value of MUSeGen by carrying out a case study where the performance of a real WMS is evaluated under workloads generated by different tools: a stress testing tool, HELP, and MUSeGen.

The rest of the article is organized as follows. Section II presents insights related to WMS and discusses related work. Our data collection methodology is presented in Section III. Section IV describes our user behavior model and

[1] https://github.com/LABORA-INF-UFG/paper-VSKA-2021

characterizes its components, whereas our workload generator is introduced in Section V. A validation of MUSeGen is presented in Section VI. Section VII presents a case study that assesses the value of MUSeGen in practice. Finally, Section VIII concludes with a summary and future work directions.

II. RATIONALE

Before presenting our data collection strategy, we provide in this section an overview of WMSs (Subsection II-A) and discuss related works (Subsection II-B).

A. OVERVIEW OF WMS

WMSs are services that deliver maps and other georeferenced information through the Web [8]. Today, Web maps are sophisticated client-server applications that dynamically combine many types of spatial data to serve users with different purposes. People interact with WMSs to publish content, to get orientation, or to develop new applications [9]. Each goal defines different ways to interact with the system. Also, clients of WMSs are designed to run either in desktop or mobile terminals. In this work, we focus on WMS users whose goal is to visit the website to get information or orientation. We also focus on users of desktop terminals, i.e., those that access the Web map through a Web browser.

Current WMSs, including Google Maps, work with multiple overlapped maps, each of them corresponding to a fixed zoom level. Each overlapped map is divided into smaller, already rendered, and discretely addressed images called *tiles* [10]. Indeed, the concept of tiles brings several advantages for the performance of a WMS. Since tiles are already rendered, they can be quickly delivered to users. Also, because they are discretely addressed, they can be cached by Internet caching services and by the user's browser. Finally, by dividing the map into tiles, when the user navigates through the map, only the new parts of the map view have to be resent from the server.

A two-dimension scheme, usually the Mercator projection [11], maps tile addresses into geospatial coordinates. In the Mercator projection, the number of tiles in both axes (X and Y) is the same and grows exponentially as the zoom level increases, following the 4^{zoom} rule: 1 tile at zoom level 0; 4 tiles at zoom level 1; 16 tiles at zoom level 2; and so one. Fig. 1 shows the division of a map into tiles up to zoom level 2 and following the 4^{zoom} rule. As we can see, the zoom levels form a pyramid of tiles. Thus, given the zoom level and the geographic coordinates, it is possible to identify the associated tile. Particularly, in Google Maps, the highest zoom level supported is 21, while the pixel resolution used for the tiles is 256×256. Thus, the highest map resolution supported by Google Maps is $256 \times 256 \times 4^{21}$ pixels.

In a desktop terminal, a WMS session starts when the user accesses the website of the map service using a Web browser. The WMS then sends the tiles needed to fill in all the space dedicated to the map on the user's screen. This space is called *bounding box*. During a session, the operations

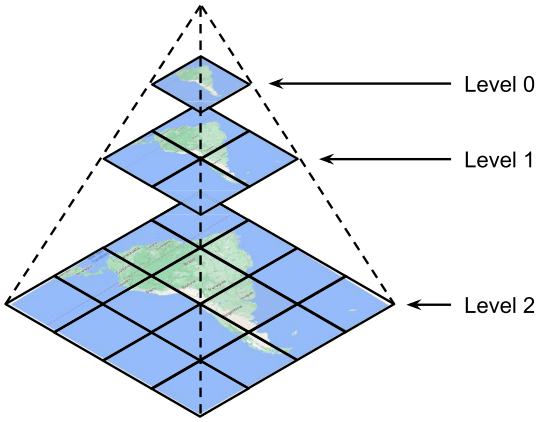


FIGURE 1. Division of a map into tiles with zoom levels 0, 1 and 2. The zoom levels form a pyramid of tiles.

commonly issued by the user are *pan, zoom-in, zoom-out, search,* and *route*. Pan and zoom are the essential operations. They allow navigation on the map, being required, respectively, for positioning the map to a region of interest and regulating the level of detail presented. The search operation allows users to search for places of interest. The route operation is employed to create paths between two or more points given a transportation mode. From the perspective of a workload model, it is important to understand how users perform these operations, how they are distributed over time, and how many tiles are requested to the server when they are executed. This knowledge is crucial to help system administrators to analyze the performance of the system and ensure a better user experience.

B. RELATED WORK

The performance of Web systems has a direct impact on the user experience. Thus, several workload models and workload generators have been proposed to characterize the workloads of different types of Web applications [3].

Barford and Crovella [12] introduced SURGE, a workload generator that employs the concept of user equivalents to generate traffic for Web servers. Krishnamurthy *et al.* [13] proposed SWAT, a workload generator that takes into account user sessions and content dependencies among requests within the same session. Calzarossa *et al.* [14] presented a comprehensive survey of the state-of-the-art in the characterization of conventional (non-geospatial) Web workloads, while Goseva-Popstojanova *et al.* [15] presented an empirical analysis of the session-based workload based on the data extracted from logs of 11 conventional Web servers. However, all these works focus on characterizing non-geospatial Web systems, in which navigation occurs mostly through hyperlinks. WMS users, on the other hand, navigate on the map by zooming and panning in the tile pyramid and, thus, access patterns for WMSs are different from those found in non-geospatial Web systems. As a consequence, conventional Web workload generators can not reproduce WMS workloads accurately [4], [5].

TABLE 1. Qualitative Comparison of Existing Works and MUSeGen.

Feature	Conventional Web workload models	RTRM model	Romoser et al.	Guan et al.	MUSeGen
Is it focused on WMSs?		✓		✓	✓
Can it generate flexible workload scenarios?	✓		✓	✓	✓
Is the work based on real data?	✓	✓	✓		✓

Traditionally, performance evaluation of WMSs is performed with stress testing tools (e.g., [16]), which implement a random tile request model (RTRM). In such models, tiles are called randomly without taking into account the relation to tiles in subsequent calls. In addition, in RTRM tools, user requests comprise one tile per time, and requests are issued with no time interval between them. Tiles are sorted according to their coordinates and tile selection is usually performed using a uniform distribution. RTRM tools allow verifying the maximum capacity of a Web server by requesting objects as quickly as possible. However, they lack the flexibility required to generate different workloads under different conditions.

Romoser *et al.* [17] analyzed the logs of the USGS EROS, a system designed for browsing and downloading Earth images. The authors present an in-depth analysis of user, image, and request characteristics of the system based on real data and, as such, are closer to our work. However, the access patterns in the EROS system are still different from those found in WMSs. For example, in EROS, the zoom operation is available through a menu, and the response time is notably higher than the response time of WMSs.

Closer to our work, Guan *et al.* [5] proposed a theoretical workload model for WMSs called HELP. This model describes how users of desktop terminals browse an online Web map and statistically characterizes map navigation behavior in relation to the pan, zoom, routing, and search operations. However, different from our work, which is based on analysis of real data gathered from real WMS users, the HELP model is based on previous findings on conventional Web workloads and empirical knowledge on the usage of WMSs.

There are also several studies focusing on strategies to improve the performance of WMSs using caching and tile prefetching algorithms [18]–[21]. The main purpose of these works is to prefetch "hot-spot" tiles and put them into a cache pool, in order to reduce the request-response time. Although tile access is highly correlated to user behavior, none of those mentioned works characterize the behavior of actual WMS users. Thus, our work contributes to such efforts and helps construct more accurate predictive tile models.

In [22], we introduced the methodology we use to anonymously collect data from users of a client application of Google Maps. In [6], we used the collected data and introduced a preliminary model of how WMS users browse a Web map. Here, we present the complete description of the model and build on top of it to develop a new workload generator for WMSs, namely MUSeGen. MUSeGen is then validated against real-world traces, and HELP. In addition, the value of MUSeGen is demonstrated through a case study involving the performance evaluation of a real WMS.

Table 1 summarizes the related work focused on workload characterization and shows aspects where our work advances the state-of-the-art.

TABLE 2. Example of URLs Generated by Google Maps.

URL	Format
URL 1	www.google.com/maps/@37.0625,-95.677068,4z
URL 2	www.google.com/maps/@37.0625,-95.677068,5z
URL 3	www.google.com/maps/search/shopping/ @37.0625,-95.677068,5z

III. DATA COLLECTION METHODOLOGY

Google Maps is a popular WMS developed by Google. Currently, Google Maps allows visualizing satellite images, traffic information, public transportation, roads, street, and other points of interest. It also allows the user to get directions, search for information on places of interest, and view streets with 360 degrees images.

Since 2014, the browser-based version of Google Maps provides information on user operations in the URLs. Particularly, in the browser-based version of the service, the URL is updated at the end of each action taken by the user on the map. To illustrate this property, consider the URLs shown in Table 2. URL 1 indicates a geographic coordinate (i.e., latitude: 37.0625, longitude: −95.677068) and the current zoom level in which the map is displayed (i.e., 4z). The coordinate and the zoom level are elementary data, and, thus, they are present in every URL. Consider now that the user performs a zoom-in operation. URL 1 will be updated to URL 2 as a response to this action. If in the next step, the user searches for the string `shopping`, URL 2 will be updated to URL 3.

Indeed, for each type of operation offered by Google Maps, the execution of that type of operation triggers the creation of a URL whose pattern is very well defined. By recording and pre-processing the generated URLs, it is possible to keep track of the actions performed by the user within a map navigation session as well as the sequence these actions are issued. It is also possible to know if the user is using Street View.

Following this approach, we developed a Google Chrome extension that anonymously collects the following data from users of Google Maps: (i) the moment the user accesses the

TABLE 3. Summary of Collected Data.

	Number of users	Number of sessions	Number of URLs
All collected data	169	2,404	120,114
Final dataset	146	1,538	36,860

Google Maps Web service; (ii) the URLs generated during the user navigation session; (iii) the timestamp at which each URL is generated; (iv) the Web page resolution in pixel, which is equivalent to the width and height of the bounding box; and (v) the moment the user leaves the Google Maps Web service.

We define a user navigation session as the time the user spends on Google Maps, which starts when she accesses the service Web page and ends when she closes the tab/window or accesses another address in the same tab. During a navigation session, the user performs several operations on the map. For each operation, Google Maps updates the URL, and an event of Google Chrome informs this update to our extension. Our extension then stores the new URL and other data (e.g., timestamp and page resolution) in the browser's database. Also, during a navigation session, different operations issued by the user result in tiles and other information (e.g., scripts and icons) being requested to the system. Our extension also collects the request-response size (in bytes) as well as the timestamp at which the request is completed. When the user ends a browsing session, our extension groups all the collected data and sends it to a server. A detailed description of our data collection methodology is presented in [6], [22].

We published our extension in the Chrome Web Store, where 169 users downloaded and installed the software, providing us with 120,114 URLs. Our extension allowed the users to consult bus time schedules in the metropolitan area of Goiania (Brazil), which corresponds to an area of nearly 7,300 km^2 and a population of more than 2.5 million people. We announced our initiative through e-mail lists and online social media. As we will show in the next session, the zoom level at which users perform the operations is an important parameter in our model. Since this information is only available in the road map, we opt for discarding in our dataset all the sessions where users browsed the Street View or the map with satellite images. After removing these sessions, we end up with a total of 146 users and 36,860 URLs, as shown in Table 3. These final set of URLs and their related data represent the dataset used in our analysis.

IV. USER BEHAVIOR IN A CLIENT APPLICATION OF GOOGLE MAPS

In this section, we first introduce our model designed to characterize user behavior in a navigation session of Google Maps (Subsection IV-A). We also discuss the main aspects considered to derive this model. Next, we present a characterization of the dataset obtained in our data collection campaign, showing the statistical distributions that best describe its main variables (Subsection IV-B). We use this dataset to parameterize our model. Finally, we discuss some limitations related to our model (Subsection IV-C).

A. PROPOSED MODEL

To model and simulate the behavior of users of Google Maps, we must take into account how users interact with the operations offered by the system. In this work, we only consider the pan, zoom, search, and route operations. We also have to consider how these interactions are distributed over time and the number of tiles requested to the server when an operation is issued. In the following, we discuss relevant aspects to be captured when modeling the behavior of WMS users.

1) DISTRIBUTION OF THE INTERACTIONS OVER TIME

We have to take into account: (i) the duration (length) of the navigation session, and (ii) the user think time. The former represents the time elapsed since the start and the end of the session. Since users can leave the page open for long periods without performing map-related activities, long sessions are not always indicative of a higher workload. Thus, it is also necessary to consider the number of actions performed by the user during the session. During a browsing session, users alternate between performing actions and interpreting the results. The time interval between these two states is called the user think time. The user think time allows to understand and simulate an important aspect of human behavior: the time needed for someone to look at the map and decide the next action. This information is important to simulate accurately how often users send requests to the system.

2) PAN OPERATION

Usually, the pan operation is performed through the drag and drop movement of the mouse. Because of that, the size and direction of the pan may vary from one movement to another. As the number of tiles requested to the server depends on the size of the pan movement, it is relevant to characterize it. The size of the pan movement can be measured in terms of the variation (in pixels) of the map position. In addition, it is a recurrent behavior to perform consecutive pan operations while positioning the map to a region of interest. The direction of such pans usually follows some logical path. This makes the probabilities of going in different directions also different. Thus, modeling these probabilities allows a more realistic emulation of the user behavior.

3) ZOOM OPERATION

Google Maps allow users to navigate between zoom levels using the mouse wheel, double click on the map, buttons on the keyboard, or specific controls in the application interface. When the initial zoom level is lower than the final, the operation is called zoom-in; otherwise, it is called zoom-out. Thus, modeling the zoom operation accurately requires knowing if it is a zoom-in or a zoom-out. It is also relevant to consider the jump size since it affects the number of tiles requested to the server. We define the jump size in a zoom operation as the (absolute) difference between the initial zoom level

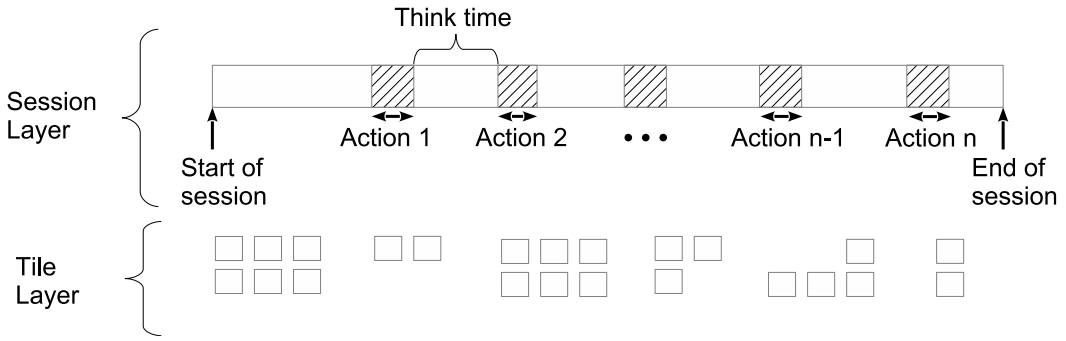


FIGURE 2. Model of the user behavior during a browsing session of Google Maps. The Session layer captures how users perform the operations offered by the system, how these operations are distributed over time, and the sequence they are issued. The Tile layer translates user actions into tile requests.

and the final zoom level. Another aspect of the zoom is that when the movement is executed using the mouse, the map is displaced. In this case, it is important to capture the direction of the movement, hereafter denoted zoom direction.

4) SEARCH AND ROUTE OPERATIONS

For the search operation, it is relevant to capture the most searched expressions and places, identifying the proportion of searches triggered by addresses, geographical coordinates, and names of places. For the route operation, it is important to characterize the distribution of the number of points belonging to the route (in systems that allow adding more than two points), the distribution of the distance between the route points, and the distribution of the selected transportation modes.

5) SEQUENCE OF OPERATIONS

In addition to modeling how users interact with the operations individually, we have to capture the sequence these operations are issued. This information allows determining the path followed during the navigation sessions. For example, a recurrent behavior may be to execute a search before a pan or a zoom.

6) REQUESTED TILES

For each operation offered by Google Maps, the model must capture the number of tiles requested to the server when this operation is executed. These tiles represent the workload in a WMS. Another aspect to be modeled is the tile popularity. This knowledge can be used, for example, to model the starting point of the navigation on a synthetic load.

Driven by the aspects discussed above, we propose a model composed of two layers, as shown in Fig. 2. The first, named Session layer, models user actions within a WMS browsing session. The second, named Tile layer, characterizes the impact of user activities on the number of tiles requested to the server and, consequently, on the volume of data transferred to the client application. For example, the Tile layer must process a pan operation and request the necessary tiles to fill in the user bounding box. Thus, the Tile layer is responsible for translating user actions into tile requests.

In the following subsection, we characterize each layer of our model by presenting the statistical distributions that best describe its variable. The final decision of what candidate distribution best fits a given variable is taken by comparing the Kolmogorov-Smirnov statistic[2] [23] and the Least Square Errors (LSE)[3] [24] of the fitted curves using a significance level of 0.05. To estimate the parameters of the distributions, we use the Maximum-Likelihood Estimation (MLE) method [25]. All the required data used in the following characterization were extracted from our dataset (described in Section III). Due to the number of users in our dataset, we opted for modeling user behavior patterns considering the aggregated behavior of the individual users, instead of grouping them into profiles and, then, characterizing each resulting group.

B. DATASET ANALYSIS AND CHARACTERIZATION

1) SESSION LAYER

In this subsection, we present the statistical analysis of the main variables that describe user actions within a session, namely, session length, user think time, size and direction of the pans, size and direction of the zooms, frequency of the operations, and sequence of issued operations.

a: SESSION LENGTH

Fig. 3 presents the empirical cumulative distribution function (CDF) of the session lengths (curve with solid line). We observe that most of the sessions are short, with the 3rd quartile of the data being approximately 4 minutes. Indeed, we see that less than 0.25% of the sessions last up more than 24 hours. This happens because some users leave the browser tab opened for days without performing any action. Since these sessions are rare and do not generate load for the server, we remove them from the dataset before modeling the session lengths. Fig. 3 also shows the fitted distribution (curve with dashed line). The Log-normal distribution with $\hat{\mu} = 4.472$ and $\hat{\sigma} = 1.778$ describes the data properly.

[2]for continuous distributions
[3]for discrete distributions

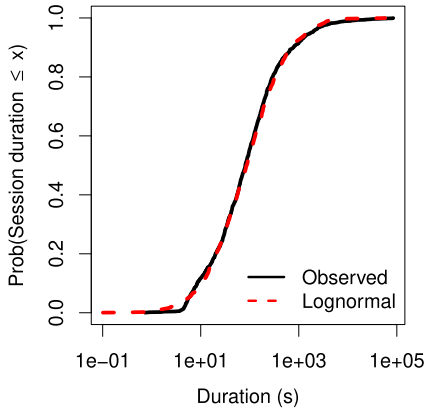


FIGURE 3. CDFs of the session lengths according to the empirical data and the best fitted distribution (Lognormal). The solid line represents the empirical data, while the dashed line represents the fitted distribution.

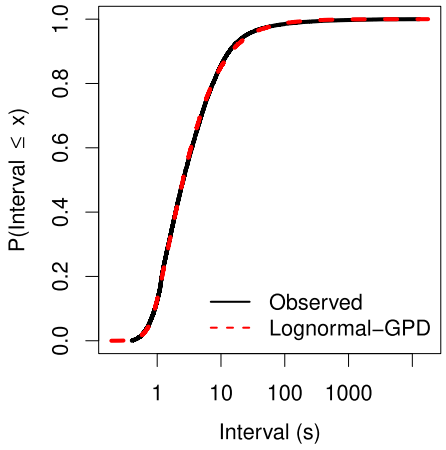


FIGURE 4. CDFs of the user think time according to the empirical data and the best fitted distribution (Lognormal-GPD). The solid line represents the empirical data, while the dashed line represents the fitted distribution.

b: USER THINK TIME

In contrast to session length, we find much higher variability in the user think time. In some cases, such variability is due to URLs generated automatically, resulting in very small time intervals between URLs. Since those time intervals do not correspond to user actions, we use a threshold to discard them. To determine a minimum threshold for the user think time, we opted for a simple approach, i.e., excluding the first percentile from the data to remove outliers [26]. This approach gives us the value of 400 milliseconds, which we use as the threshold.

Fig. 4 presents the empirical CDF of the user think time after removing the samples below the threshold (curve with solid line), as well as the fitted distribution (curve with dashed line). We find that a mix of the Log-normal and the Generalized Pareto Distribution (GPD) is the best choice for our data. We use a right truncated Log-normal with $\hat{\mu} = 0.668$ and $\hat{\sigma} = 0.577$ to model the distribution body up to a threshold $\hat{\theta} = 1.107$, and the GPD with $\hat{\tau} = 2.166$ and $\hat{\xi} = 0.895$ to model the tail from $\hat{\theta}$.

c: PAN OPERATION

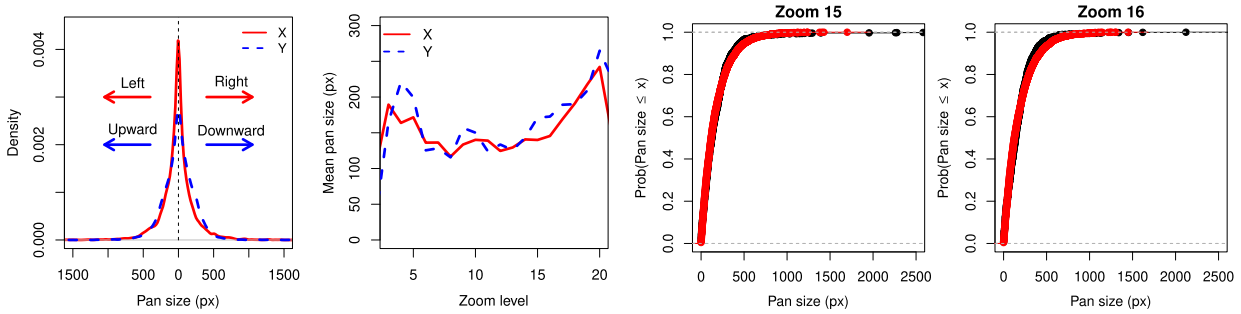
Fig. 5(a) shows the distribution of the size of the pan movements, in both X- and Y-axis, in our dataset. In the X-axis (curve with a solid line), values on the left and on the right of 0 represent, respectively, left and right movements. In the Y-axis (curve with a dashed line), they indicate, respectively, upward and downward shifts. We observe that the majority of the movements are small and, in general, the movements on the Y-axis are larger. We conjecture that this happens because most computer monitors today operate in wide-screen format [27], which displays less information on the Y-axis. Thus, there is a need for larger movements in the vertical direction. As shown in the figure, the size of a pan can be equal to 0 in one of the axes. This means a movement in a single direction (horizontal or vertical) and usually occurs when the keyboard arrow keys are used to move on the map.

We also analyze the mean size of the pan movements per zoom level. Fig. 5(b) shows this measurement for both X- and Y-axis. In both axes, the mean size of the pan movements tends to increase as the zoom level increases. We compute the Pearson coefficient between the two metrics and confirm, in both axes, the existence of a positive correlation between them. In view of this, we opt for modeling the distribution of the size of the pan movements as a function of the zoom level. We find that, despite variations in parameters, the Geometric is the best choice for describing the distributions on both axes for all values of zoom. We also observe that as the zoom level (z) increases, the parameter p of the Geometric distributions decreases. Applying a linear regression method to these metrics (z and p), we obtain Equations 1 and 2. These equations allow us to obtain the Geometric distributions of the size of the pan movements for a given z in axes X and Y, respectively. Fig. 5(c) illustrates the CDFs of the empirical distributions of the size of the pan movements, in the Y-axis, for zoom levels 15 and 16 extracted from our dataset. The figure also shows the CDFs of the fitted Geometric distributions obtained from Equation 2. We observe that the fitted distributions describe our data satisfactorily. This is also observed for other zoom levels for both axes.

$$p_x = -0.0001376 \times zoom\_level + 0.0079955 \quad (1)$$

$$p_y = -0.0002038 \times zoom\_level + 0.0085857 \quad (2)$$

Finally, we investigate patterns related to the direction of the pans. Considering the two axes, we obtain 8 movement possibilities: Left and Up (L,U), Left and Down (L,D), Left and 0 (L,0), Right and Up (R,U), Right and Down (R,D), Right and 0 (R,0), 0 and Up (0,U), and 0 and Down (0,D). We model these possibilities as states and compute the probability of changing from one state to another using our dataset. Table 4 illustrates the transition matrix derived from our data. We noticed that successive pans have a high probability of being in the same direction.



(a) Size of the pan movements (b) Mean pan size per zoom level. (c) CDFs of the empirical and fitted Geometric distributions for the pan size

FIGURE 5. Pan size analysis. (a) Size of the pan movements in both X- and Y-axis. The curve with a solid line represents the size of the movements (in pixel) in the X-axis, while the curve with a dashed line represents the size of the movements (in pixel) in the Y-axis. (b) Mean size of the pan movements per zoom level in both X- and Y-axis. The values are given in pixels. (c) CDFs of the empirical distributions of the size of the pan movements, in the Y-axis, for zoom levels 15 and 16 (red curves) and CDFs of the corresponding fitted Geometric distributions (black curves).

TABLE 4. Transition Matrix for the Pan Directions.

	(0,D)	(0,U)	(R,0)	(R,D)	(R,U)	(L,0)	(L,D)	(L,U)
(0,D)	0.269	0.135	0.038	0.135	0.077	0.019	0.173	0.154
(0,U)	0.190	0.190	0.000	0.086	0.138	0.017	0.155	0.224
(R,0)	0.111	0.056	0.056	0.222	0.222	0.000	0.167	0.167
(R,D)	0.006	0.002	0.002	0.437	0.144	0.004	0.182	0.222
(R,U)	0.003	0.010	0.005	0.151	0.437	0.002	0.220	0.172
(L,0)	0.087	0.087	0.087	0.130	0.087	0.130	0.174	0.217
(L,D)	0.007	0.006	0.003	0.175	0.192	0.002	0.464	0.151
(L,U)	0.002	0.010	0.002	0.210	0.154	0.005	0.169	0.449
Begin	0.010	0.009	0.005	0.238	0.225	0.005	0.268	0.240

d: ZOOM OPERATION

Fig. 6(a) shows the frequency in which each zoom level is accessed in our dataset. The frequency of access differs significantly for each zoom level. For example, levels between 13 and 17 are the most accessed. This is so because they allow comfortable navigability while providing a satisfactory level of detail. Levels higher than 17 are less accessed due to the proximity to the planet's surface, making small the region shown on the screen. On the other hand, levels lower than 13 lack details, explaining the gradual drop in their access. Similar results were reported in [28] when analyzing traces of map servers in Spain. In our dataset, zoom level 4 is more accessed than their neighbors. This happens because this is the default level for beginning the navigation in most of the collected sessions.

Next, we analyze the size of the zoom jumps. Fig. 6(b) shows the distribution of this variable. Values on the left (in black) of the distribution represent jumps generated by performing zoom-out, whereas values on the right (in red) are generated by zooming-in. We find that the vast majority of the zoom jumps are small, with the 3rd quartile of the data set to 2 and the 99 percentile set to 7. We also observe that the size of the jumps varies according to the zoom level, as shown in Fig. 6(c). At lower zoom levels, the probability of jumps due to zoom-in is higher, whereas, at higher zoom levels, the jumps by zooming-out are more likely.

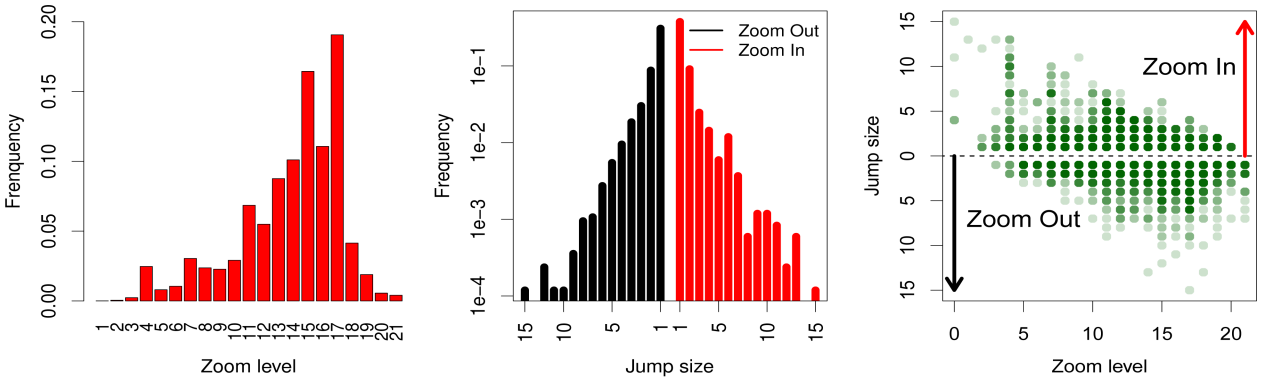
Fig. 7 shows the empirical CDFs of the size of the movements of the central coordinate for both axes. These movements are caused by repositioning the map due to a zoom operation. Values are given in pixel. The figure also shows the CDFs of the fitted distributions. We observe that the Geometric distribution with parameter $p = 0.0138865319$, shown in Fig. 7(a), fits our data satisfactorily in the X-axis, while the Geometric distribution with parameter $p = 0.0086910695$, illustrated in Fig. 7(b), best represents our data in the Y-axis.

Finally, when the map is repositioned due to a zoom operation, the movement of the central coordinate may follow different directions. Considering the two axes, we obtain 9 movement possibilities: Left and Up (L,U), Left and Down (L,D), Left and 0 (L,0), Right and Up (R,U), Right and Down (R,D), Right and 0 (R,0), 0 and Up (0,U), 0 and Down (0,D), and (0,0). Unlike the pan, here we can have a movement of 0 pixels on both axes, meaning that the movement does not change the central coordinate of the map. This happens when the zoom operation is not performed using the mouse. We model these possibilities as states and compute the probability of changing from one state to another using our dataset. Table 5 illustrates the transition matrix derived from our data. We observe that successive zooms have a high probability of being performed in the same direction.

e: ISSUED OPERATIONS

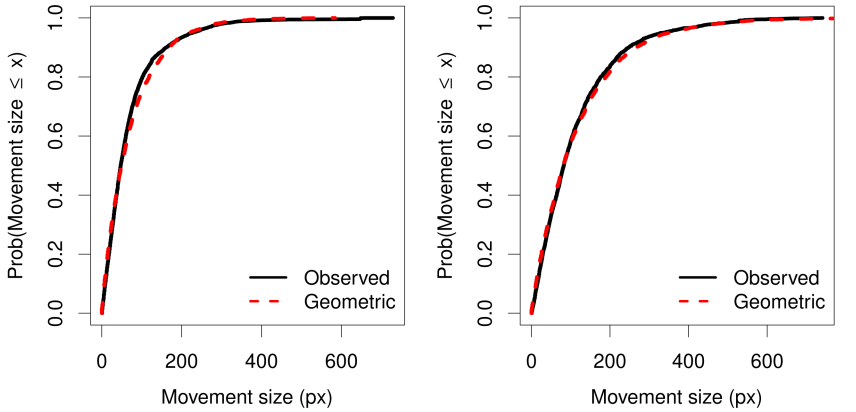
The last variable of the Session layer characterizes the relative frequency of the considered operations, i.e., pan, zoom, search, and route. The result is shown in Fig. 8(a). We observe that together, the pan and the zoom represent 84% of the total actions performed on the map. The route operation is more frequent than the search. This happens because the route is composed of two or more steps: a starting point, an ending point, and any number of intermediate steps. Since each step consumes resources from the server, we opt for counting them separately.

We also observed correlations between the zoom level and the frequency of access of some operations, especially the



(a) Zoom level frequency (b) Zoom jump sizes (c) Zoom jump sizes per zoom level

FIGURE 6. **Zoom analysis. (a) Frequency in which each of the 21 zoom levels is accessed. (b) Distribution of the size of the zoom jumps. Values on the left of the distribution (in black) represent jumps generated by zooming-out, whereas values on the right (in red) are generated by zooming-in. (c) Size of the zoom jumps per zoom level. Values above 0 represent jumps caused by zooming-in, whereas values below 0 correspond to jumps caused by zooming-out.**



(a) Empirical and fitted Geometric distribution for the X-axis (b) Empirical and fitted Geometric distribution for the Y-axis

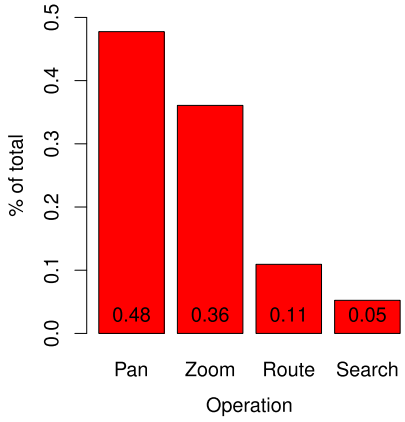
FIGURE 7. **CDFs of the size of the movements of the central coordinate according to the empirical data and the best fitted distribution (Geometric) in both axes. The solid lines represent the empirical data, while the dashed lines represent the fitted distributions.**

TABLE 5. **Transition Matrix for the Zoom Directions.**

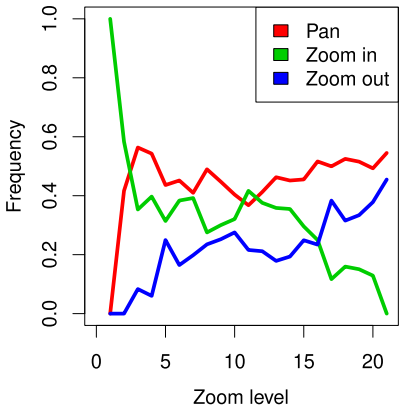
	(0,0)	(0,D)	(0,U)	(R,0)	(R,D)	(R,U)	(L,0)	(L,D)	(L,U)
(0,0)	0.855	0.000	0.000	0.000	0.034	0.034	0.000	0.026	0.051
(0,D)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
(0,U)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
(R,0)	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
(R,D)	0.015	0.000	0.000	0.000	0.418	0.088	0.000	0.086	0.394
(R,U)	0.008	0.000	0.000	0.000	0.076	0.481	0.000	0.339	0.095
(L,0)	0.000	0.000	0.000	0.000	0.200	0.400	0.200	0.200	0.000
(L,D)	0.027	0.000	0.000	0.000	0.080	0.335	0.002	0.436	0.119
(L,U)	0.008	0.000	0.000	0.000	0.260	0.113	0.002	0.094	0.525
Begin	0.058	0.001	0.001	0.001	0.207	0.233	0.002	0.249	0.248

zoom-in, zoom-out, and pan. Fig. 8(b) shows the relative frequency of such operations per zoom level. We observe that as the zoom level increases, the number of issued pans and zoom-out tends to increase, while the number of zoom-in tends to decrease.

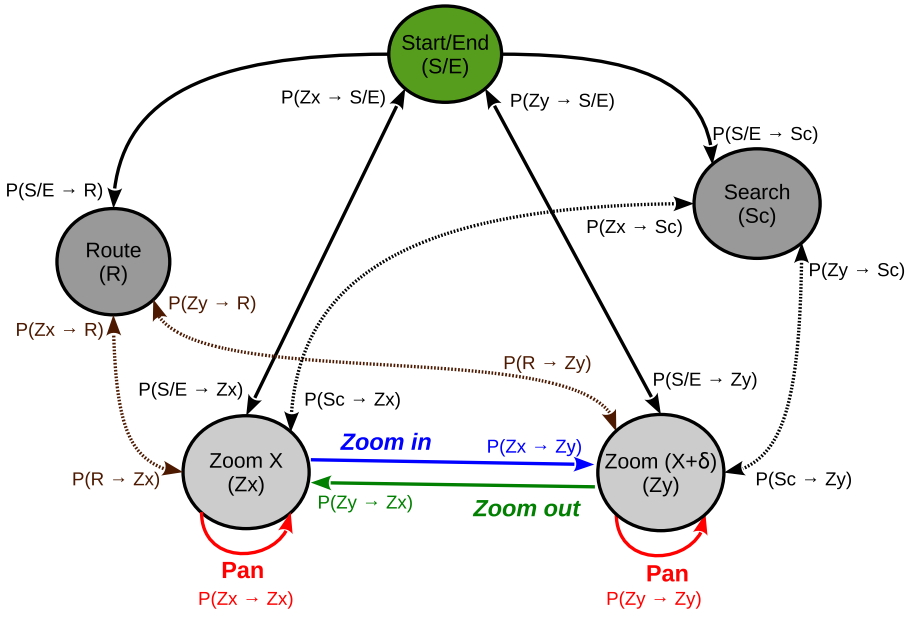
Finally, we capture the navigation pattern within a Google Maps session using a state diagram, where states represent the system operations and transitions (arcs), connecting the states, reflect the probability of the next operation. However, to reduce the number of states, we employ a slightly different configuration. In our model, the zoom levels and the route and search operations are represented as states. On the other hand, due to the correlations illustrated in Fig. 8(b), the pan and zoom (in and out) are modeled as transitions between zoom



(a) Relative frequency of operations



(b) Relative frequency of operations per zoom level



(c) State diagram

FIGURE 8. Issued operation analysis. (a) Relative frequency of each operation. (b) Relative frequency of pans and zooms (in and out) per zoom level; (c) Part of the state diagram derived from our dataset, showing the `Search`, `Route`, and two zoom level states. `Star/End` state is represented in green. The zoom levels as well as the route and search states are represented in gray. Pan and zoom are modeled as transitions between zoom levels. Pans are represented in red, zoom-in in blue, and zoom-out in green. `P(S1 → S2)` represents the probability of changing from state `S1` to state `S2`.

levels. Transitions within the same zoom level represent the pan, while arcs between different zoom levels represent the zoom. In addition, if the transition from zoom level X to zoom level Y is such that $X < Y$, then the arc represents the zoom-in operation. If $X > Y$, then it models the zoom-out. Fig. 8(c) illustrates part of the state diagram derived from our dataset, showing the `Search` and `Route` states and two zoom level states. The whole state diagram is omitted for the sake of clarity. The `Start/End` state represents the start/end of the navigation session. The transitions from this state are linked to the zoom level states, representing that the navigation may start and end at any zoom level. There are also transitions between the `Start/End` state and the `Search` and `Route` states. These transitions represent the cases where the first access is made through URLs that direct the WMS to the results of the search and route operations. The probability of changing between states, represented by `P(S1 → S2)` in Fig. 8(c), with `S1` and `S2` representing states, is extracted from our dataset. However, due to space constraints, we omit the state transition matrix.

2) TILE LAYER

In this subsection, we analyze the tiles requested to Google Maps servers to satisfy the user operations. Since the zoom and the pan correspond to more than 80% of the operations in a WMS, our analysis focuses on them. Particularly,

we analyze the number of tiles requested to the server to fill in the bounding box when a pan or zoom operation is executed. This number, in turn, is directly related to the user's computer screen resolution.

Fig. 9(a) illustrates the distribution of the number of tiles returned to satisfy a zoom operation in our dataset. We observe that the values of 18, 24, and 28 are the most returned tile quantities. Indeed, these are the quantities needed to fill in the bounding box in the 1366 × 768 screen resolution, which was the most world-wide used resolution at the time of our data collection [27]. Depending on how tiles are organized, this resolution can support 6 or 7 tiles on the X-axis and 3 or 4 tiles on the Y-axis, resulting in 18, 21, 24, or 28 tiles in total. Tile quantities of 40 and 45 are also very frequent and are related to the 1920 × 1080 screen resolution, the second resolution most commonly used [27].

Fig. 9(b) shows the distribution of the number of tiles returned to satisfy a pan operation in our dataset. The most returned tile quantity is 0 since most of the pan movements are of small sizes. Thus, these movements show tiles that had already been brought from the server. Similarly, the probability of requesting a high number of tiles to satisfy a pan is small, since larger pan movements are rare.

Using the information on the number of tiles returned by the server, we estimate the volume of data transferred per operation (pan and zoom). We find that 99% of the pans require the transfer of less than 344.71 KB of tile images, while 99% of the zooms need less than 883.71 KB.

C. DISCUSSION

Our user behavior model has several components. At the Session layer, user behavior is represented in terms of session length, user think time, and issued operations. At the Tyle layer, the model represents the workload generated when a given operation is performed.

While we have designed the WMS user behavior model and characterized its parameters using data from desktop terminals, we believe that the methodology applied in this work can also be used to build workload models for mobile terminals. This is so since the aspects captured in our model (e.g., distribution of the interactions over time, pan operation, zoom operation, the sequence of operations, and the number of tiles requested to the server) are also relevant for describing mobile user behavior. However, the parameters that characterize the components of the model vary in a mobile terminal. For example, the amount of tiles requested per pan or zoom depends on the user's computer screen resolution, which may be different in a mobile terminal. Due to these slight differences, modeling WMS user behavior of mobile terminals is valuable. This is an interesting direction for future investigation.

Moreover, since our data collection campaign focused on Google Maps, minor adaptations may be necessary to adjust our model to other WMS. An example is the highest zoom level supported, which may be different in other WMSs.

Algorithm 1 Emulating a User Session Using MUSeGen

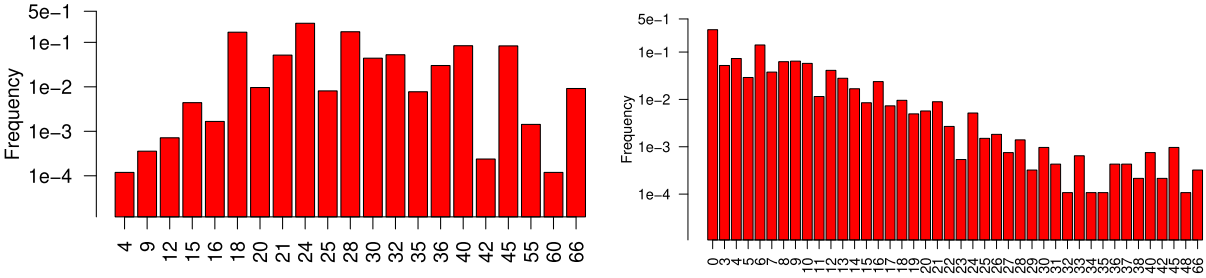
Output: webMapUserSession – set of actions and intervals that define a session of a web map user.

1 *start_point* ← Sample(Tile popularity distribution)
2 *init_zoom* ← Sample(Probability of 'start/end' state)
3 *screen_size* ← Sample(SGS)
4 *op.type* ← *start_session*
5 *op.params* ← *start_point*, *init_zoom*, *screen_size*
6 webMapUserSession.addAction(*op*)
7 *op.type* ← Sample(State Diagram ('start/end'))
8 **while** *op.type* ≠ *'Start/End'* **do**
9 *time_int* ← Sample (User think time distribution) `// Lognormal-GPD in Figure 4`
10 webMapUserSession.addInterval(*time_int*)
11 **if** *op.type* = *pan* **then**
12 *pan_size* ← Sample(Pan size distributions)
13 `// Geometric distributions`
14 *pan_direction* ← Sample(Pan transition matrix)
15 *op.params* ← *pan_size*, *pan_direction*
16 **end**
17 **else if** *op.type* = *zoom* **then**
18 *center_pt_ch* ← Sample(Zoom distribution)
19 `// Geometric distributions in Figure 7`
20 *zoom_direction* ← Sample(Zoom transition matrix)
21 *op.params* ← *center_pt_ch*, *zoom_direction*
22 **end**
23 webMapUserSession.addAction(*op*)
24 *op.type* ← Sample(State Diagram (*op.type*))
25 **end**
26 *time_int* ← Sample(User think time distribution)
27 webMapUserSession.addInterval(*time_int*)
28 *op.type* ← *end_session*
29 webMapUserSession.addAction(*op*)`// Last action`

V. MUSeGen: A WORKLOAD GENERATOR FOR WMS

Based on the analysis presented in Section IV, we implement MUSeGen, a workload generator for WMSs. MUSeGen uses the state diagram, described in Subsection IV-B1, as well as the distributions that characterize the zoom, the pan, and the user think time to model user actions within a WMS session. Since our data do not provide enough information for modeling the operations of search and route, they are not produced by this version of the workload generator.

Algorithm 1 describes the main steps performed by MUSeGen to emulate a user session. First, the algorithm starts the session by setting up an initial navigation point, an initial zoom level, and the user's screen size (lines 1-6). The initial navigation point can be drawn from the tile popularity distribution that describes the access to the tile server content (line 1). The initial zoom level is randomly chosen considering the output probabilities of the `Start/End`



(a) Returned tiles due to a zoom (b) Returned tiles due to a pan

FIGURE 9. Tile analysis. (a) Distribution of the amount of tiles returned due to a zoom operation. (b) Distribution of the amount of tiles returned due to a pan operation.

state of the state diagram (line 2). The user's screen size, required to compute the number of tiles to fill in the bounding box, is sampled from the distribution of the most widely used monitor resolutions according to StatCounter Global Stats (SGS) [27] (line 3).

We use the state diagram derived in Section IV-B1 as the main engine of our workload generator. The algorithm iterates over the states (lines 8-25) according to the transition probabilities. This iteration process finishes when the execution reaches the `Start/End` state (line 8).

Inside the loop, the algorithm identifies and processes the operation (lines 9-22). Overall, three types of operations can be randomly chosen. If the chosen operation is the pan (lines 11-16), the algorithm selects the pan direction from the pan transition matrix. It also selects the size of the pan (in both X- and Y-axis) using the Geometric distributions of the size of the pan movements. If the chosen operation is the zoom (lines 17-22), then the algorithm selects the size of the movement of the central coordinate and the direction of the movement by sampling, respectively, from the fitted Geometric distributions and from the zoom transition matrix. Finally, if the chosen operation is the `Start/End` state, then the algorithm exits the loop. Between two actions there is a random user think time, as characterized by the Lognormal-GPD distribution shown in Figure 4 (line 9). When the execution exits the loop, the algorithm generates the last user think time (lines 26-27) and closes the user session (lines 28-29).

The output of the algorithm is a list of user actions interleaved by time intervals. This list must be passed to a tile server in order to generate the workload effectively. Thus, in MUSeGen, sessions are generated offline and the effective workload generation occurs later. Since the effective workload generation of each session may consume some time, this approach avoids introducing an error that could vary notably as a function of the hardware resources (e.g., the speed of the memory and CPU) and the software used for creating the HTTP requests.

The computational complexity of Algorithm 1 is defined by the `while` loop (lines 8–25), which lasts until the `Start/End` state being reached. In each iteration, this loop produces a single operation and the user think time. In other words, the computational complexity is $\mathcal{O}(n)$, where n is the number of operations produced by the loop. The number of operations in each algorithm execution is defined by a stochastic model represented by the state diagram shown in Fig. 8(c). Thus, the exact number of operations is unknown a priori, but we can derive an estimation based on the empirical data used to build the model. As we previously described, the largest user session has no more than 24 hours and the smallest user think time is not less than 400 milliseconds. Thus, the probability of the stochastic model generates more than 216,000 ($= 24 \times 60 \times 60 \times 1000/400$) operations is negligible. In addition, the generation of sessions of different users is independent. This provides high scalability to our algorithm since the sessions can be generated concurrently.

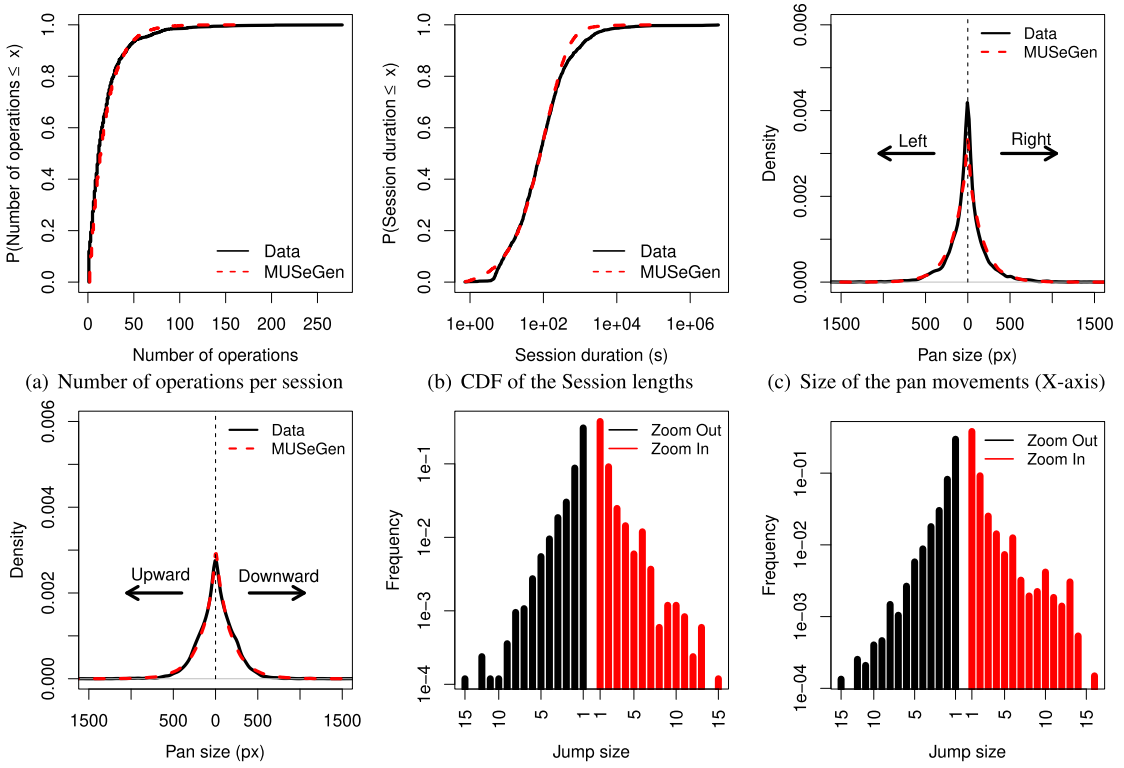
VI. VALIDATION OF MUSeGen

In this section, we present a validation of MUSeGen. The purpose is to (i) verify if MUSeGen produces realistic workloads (Subsection VI-A), and (ii) assess how different the user behavior patterns produced by MUSeGen and HELP are (Subsection VI-B).

A. HOW REALISTIC ARE THE USER BEHAVIOR IN MUSeGen?

To assess if MUSeGen can reproduce the behavior of real WMS users, we compare statistical characteristics of the synthetic traces generated by our tool with those found in real data. Particularly, we compare metrics that are not explicitly encoded in MUSeGen state diagram – but can be derived from it – against real data obtained from our dataset. Examples of such metrics are the number of operations per session, session length, pan sizes, and zoom jump sizes.

The distribution of the number of operations per session could be extracted from our dataset. However, in MUSeGen, this metric is obtained indirectly by the state diagram. Fig. 10(a) shows the CDF of this metric obtained from the state diagram and the one extracted empirically from our dataset. We can see that the two distributions are tightly close to each other, with the empirical distribution presenting a larger tail.



(a) Number of operations per session (b) CDF of the Session lengths (c) Size of the pan movements (X-axis)

(d) Size of the pan movements (Y-axis) (e) Distribution of the size of the jumps - dataset (f) Distribution of the size of the jumps - MUSeGen

FIGURE 10. **MUSeGen validation. (a) CDF of the number of operations per session generated by MUSeGen (dashed line) and found in our dataset (solid line). (b) CDF of the session lengths generated by MUSeGen (dashed line) and found in our dataset (solid line). (c) Distribution of the size of the pan movements in the X-axis generated by MUSeGen (dashed line) and found in our dataset (solid line). (d) Distribution of the size of the pan movements in the Y-axis generated by MUSeGen (dashed line) and found in our dataset (solid line). (e) Distribution of the zoom jump sizes found in our dataset. Values on the left of the distribution represent jumps due to zoom-out, while values on the right are due to zoom-in. (f) Distribution of the zoom jump sizes generated by MUSeGen. Values on the left of the distribution represent jumps due to zoom-out, while values on the right are due to zoom-in.**

Similarly, session lengths in MUSeGen are not derived directly from real data but depend on the number of operations generated by the state diagram and the time intervals sampled from the user think time distribution. Fig. 10(b) illustrates the CDF of the session lengths generated by MUSeGen and the one found in our dataset. We can see that the distribution of the session lengths produced by MUSeGen captures the overall trend in actual data, with the latter presenting a larger tail.

Pan movements in MUSeGen are generated using two types of information: the distribution of the size of the pans and the pan directions. The combination of these two variables results in the distributions shown in Figs. 10(c) and 10(d), representing the pan movements in axes X and Y, respectively. These figures also illustrate the empirical distributions of the pan movements obtained from our dataset. We can see that the fitted distributions for both axes match closely with the actual data, with the distribution for the Y-axis performing better than that for the X. We also observe that the transition matrix used to model the pan directions is effective, since similar to the actual data, the distribution of the size of the pans generated by MUSeGen is symmetric to 0.

TABLE 6. **Characteristics of Empirical and Synthetic Data.**

Metrics	MUSeGen	Dataset
Mean number of issued operations	18.62	18.55
Mean session lengths	239.59	655.85
Mean pan size - X-axis	-9.48	-6.09
Mean pan size - Y-axis	8.41	3.93
Mean zoom jump size	0.244	0.091

Finally, the state diagram in MUSeGen is responsible for choosing the zoom levels at which the operations are accessed. This means that the empirical distribution of the zoom level accesses and the empirical distribution of the zoom jump sizes are not directly used in our model. Fig. 10(f) shows the zoom jump sizes obtained synthetically using MUSeGen. In comparison with the empirical distribution (presented in Fig. 10(e)), we can observe that the distribution generated by the state diagram shows similar patterns to those found in our dataset.

Table 6 provides statistical summaries of the metrics evaluated in this subsection for both the synthetic traces and our dataset. The results illustrated in the table, and also obtained

TABLE 7. The HELP and Custom HELP Models. In the Table, Z_{curr} Represents the Current Zoom Level and Z_{max} the Maximum Zoom.

	HELP		Custom HELP	
Variable	**Distribution/Value**	**Parameters**	**Distribution/Value**	**Parameters**
Session length	Weibull	$k = 0.311$; $\alpha = 0.2912$	Same as HELP	$k = 0.925902$; $\alpha = 17.83891$
User's think time	Pareto	$k = 1$; $\alpha = 1.5$	Same as HELP	$k = 0.4$; $\alpha = 1.483283$
Freq. of operations	Equations 3, 4, 5	$\omega = 0.9$; $\lambda = 3$	same as HELP	$\omega = 0.426925$; $\lambda = 0.45756$
Pan direction	Uniform	$a = 0;\ b = 3$, with 0 - Up 1 - Down 2 - Right 3 - Left	same as HELP	same as HELP
Pan size	1 *tile* = 256 px	-	same as HELP	same as HELP
Screen resolution	3x3 *tiles* = 768x768 px	-	same as HELP	same as HELP
Zoom jump size	Uniform	$a = 1$; $b = \mid Z_{curr} - Z_{max} \mid$ (zoom in) $b = Z_{curr}$ (zoom out)	same as HELP	same as HELP

from the graphical inspection, show that traces generated by MUSeGen present statistical characteristics very close to those found in our dataset. Thus, there is strong evidence that MUSeGen can reproduce realistic WMS user behavior.

B. HOW DIFFERENT ARE THE USER BEHAVIOR PATTERNS PRODUCED BY MUSeGen AND HELP?

In this subsection, we assess if user behavior patterns produced by MUSeGen are significantly different from those produced by HELP. Thus, the validation presented herein illustrates the value of MUSeGen.

To contrast both models, we implement two versions of a workload generator based on HELP. The first version, hereafter called HELP, uses the same parameterization as the one provided in [5]. The second version, hereafter called Custom HELP, is parameterized using our dataset, i.e., the empirical data also used in the parameterization of MUSeGen.

Table 7 summarizes the main characteristics of HELP as described in [5]. The frequency of operations per zoom level is modeled using Equation 3 for the zoom-in, Equation 4 for the zoom-out, and Equation 5 for the pan.

$$f(l;\lambda) = 1 - \left(\frac{l}{l_{max}}\right)^{\lambda}, \tag{3}$$

$$f(l;\omega,\lambda) = \omega\left(\frac{l}{l_{max}}\right)^{\lambda}, \tag{4}$$

$$f(l;\omega,\lambda) = (1-\omega)\left(\frac{l}{l_{max}}\right)^{\lambda}, \tag{5}$$

where l is the desired zoom level and l_{max} is the maximum zoom level.

Table 7 also summarizes the main characteristics of Custom HELP, where the HELP model (more specifically, the session lengths, user think time, and the frequency of operations per zoom level) is parameterized using our dataset. To estimate parameters ω and λ of Equations 3, 4, and 5 using our dataset, we use a non-linear regression model, resulting in three different values for each parameter. We then compute the mean of the three values to obtain the final estimate for each parameter.

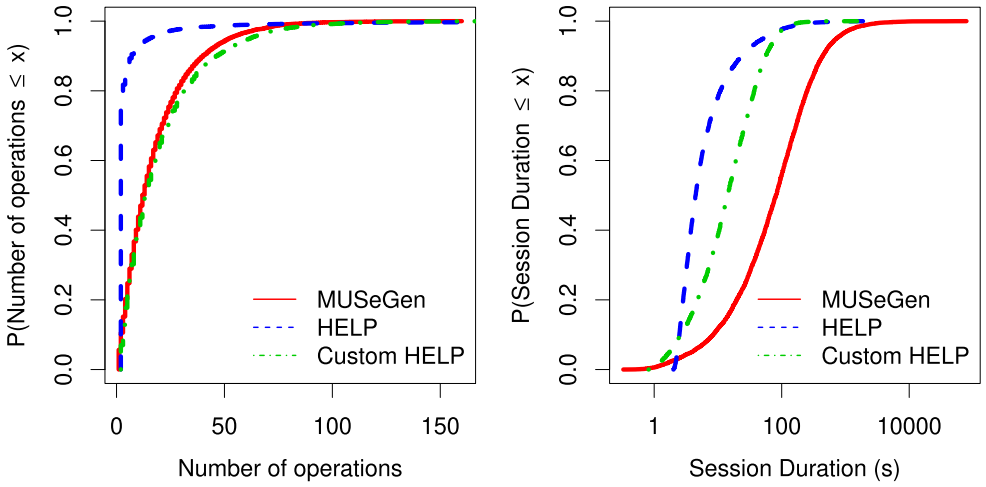
We then simulate the list of user operations generated by MUSeGen, HELP, and Custom HELP and compare the three models concerning the number of issued operations, session lengths, pan sizes, frequency of operations per zoom level, and the number of requested tiles per pan and zoom. To choose the initial point of the simulated navigation sessions, we use a dataset of the estimated population of cities in the USA in the year 2000.[4]

Fig. 11(a) depicts the CDF of the number of operations issued per session in each model. We can see that MUSeGen and Custom HELP sessions tend to perform more operations than the HELP ones. Indeed, on average, the number of issued operations per session in MUSeGen and Custom HELP is nearly four times the one in HELP. This happens because the parameterization proposed by HELP for this metric derives from studies on the number of performed operations in traditional Web sites, which do not present the interactivity of a Web map.

The CDFs of the session lengths are depicted in Fig. 11(b). On average, MUSeGen and Custom HELP sessions last longer than the HELP ones. This is expected since MUSeGen and Custom HELP sessions perform more operations than the HELP sessions. On average, MuseGen sessions are longer than the Custom HELP ones because of the different distributions that model user think time in each model.

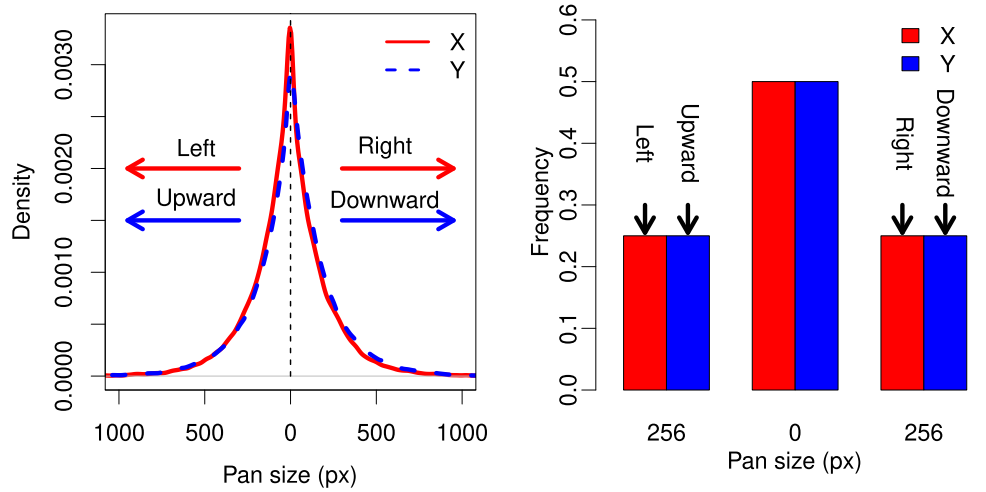
Figs. 12(a) and 12(b) show the patterns of the pan sizes in MUSeGen and HELP/Custom HELP, respectively. Since Help and Custom HELP use the same distribution and parameterization for the pan size, they follow the same pattern illustrated in Fig. 12(b). We observe that the pan sizes in

[4]http://www.baruch.cuny.edu/geoportal/data/esri/esri_usa.htm



(a) CDFs of the number of issued operations (b) CDFs of the session lengths

FIGURE 11. Comparison of MUSeGen, HELP, and Custom HELP in relation to (a) the distribution of the number of issued operations; and (b) the distribution of the session lengths.



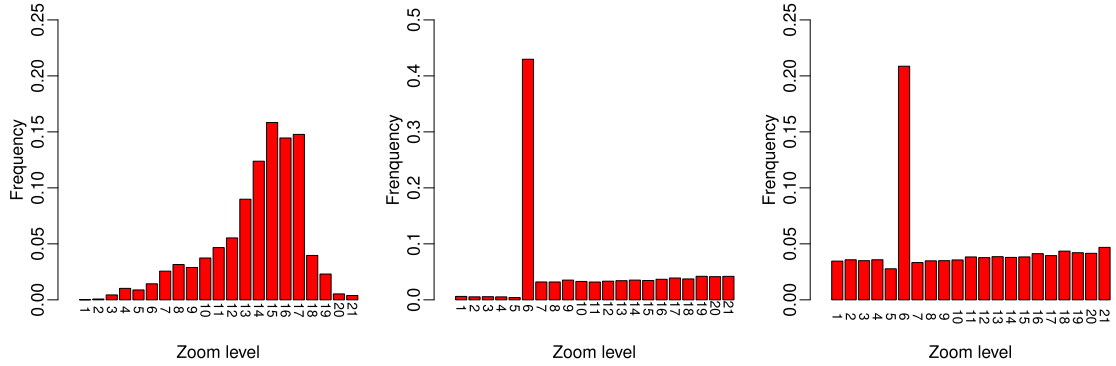
(a) MUSeGen (b) HELP/Custom HELP

FIGURE 12. Comparison of MUSeGen and HELP/Custom HELP in relation to the distribution of the size of the pan movements (in pixel) in both X- and Y-axis.

MUSeGen and HELP/Custom HELP are significantly different. In HELP/Custom HELP the pan size is fixed and is carried out in only four directions (left, right, up, and down). MUSeGen, on the other hand, varies not only the pan sizes but also the pan directions, which consists of eight possibilities as described in Section IV-B1.

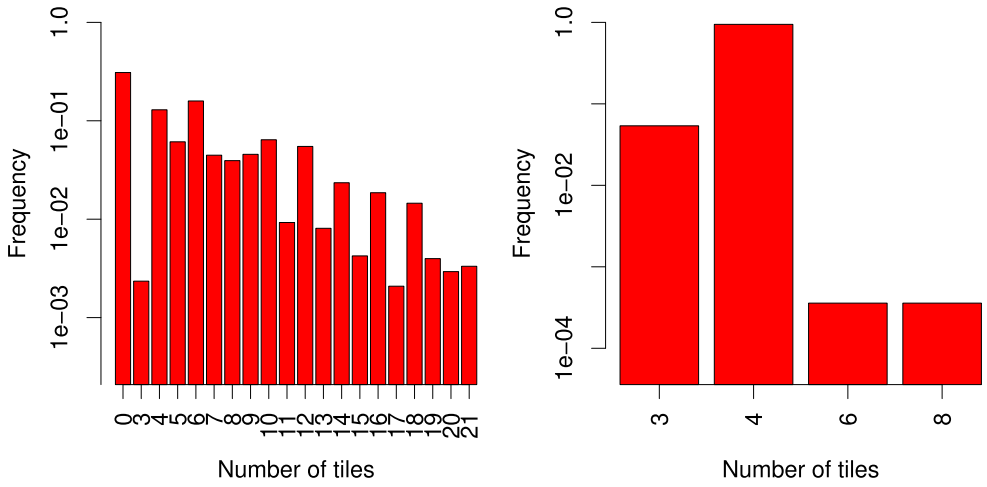
Figs. 13(a), 13(b), and 13(c) show the frequency the zoom levels are accessed in MUSeGen, HELP, and Custom HELP, respectively. In MUSeGen, the frequency of access increases up to zoom levels 15 to 17 and then decreases at higher zoom levels. As mentioned before, similar results were reported by another study on map servers [28]. In HELP, the frequency of access is high at zoom level 6. This happens because HELP does not model a distribution of zoom levels for the initial access. Then, zoom level 6 is the default initial zoom level of all sessions [5]. Custom HELP presents patterns similar to HELP, but with lower frequency. This is so because Custom HELP sessions are longer than HELP, and the simulated user has a higher chance to navigate to other zoom levels. Moreover, except for zoom level 6, in HELP and Custom HELP, the frequency of access tends to grow as the zoom level grows.

Finally, we analyze the behavior of the models concerning the number of requested tiles. Figs. 14(a) and 14(b) show the amount of tiles requested per pan in MUSeGen and HELP/Custom HELP, respectively. Since Help and Custom HELP use the same parameterization for the user's screen resolution, they follow the same pattern illustrated in Fig. 14(b). We can see that the number of requested tiles varies significantly more in MUSeGen. This happens because MUSeGen



(a) MUSeGen (b) HELP (c) Custom HELP

FIGURE 13. Comparison of MUSeGen, HELP, and Custom HELP in relation to the frequency the zoom levels are accessed.



(a) MUSeGen (b) HELP/Custom HELP

FIGURE 14. Comparison of MUSeGen, HELP, and Custom HELP in relation to the number of tiles requested per pan.

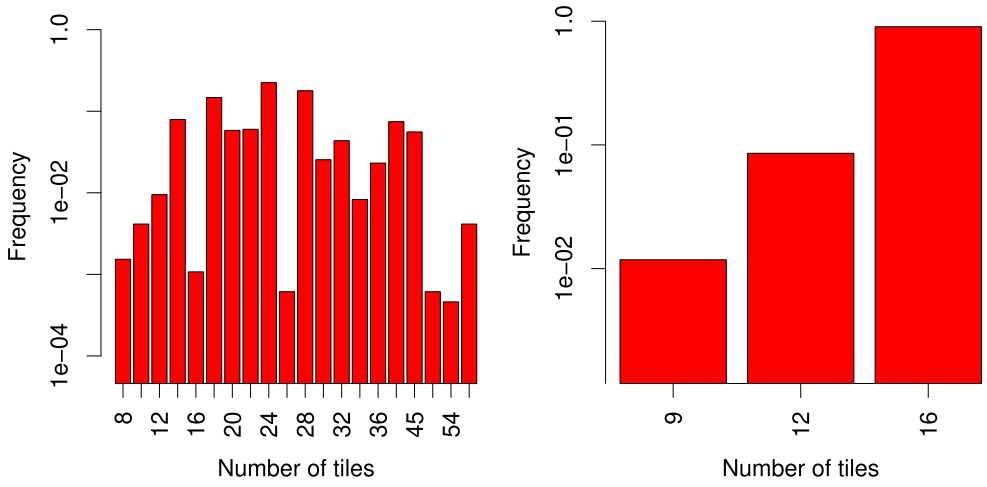
considers not only multiple pan sizes but also multiple user's screen resolutions. HELP, on the contrary, uses fixed values for the pan size and the screen resolution. This result shows the impact of the pan sizes and the screen resolution on the requests sent to the tile server. We also find that, approximately 90% of MUSeGen sessions request up to 500 tiles, while the largest number of tiles requested per session is 1500. In HELP and Custom HELP, these numbers are 200 and 800 tiles, respectively. Indeed, on average, MUSeGen sessions request twice more tiles than HELP/Custom HELP ones.

Figs. 15(a) and 15(b) show the distribution of the number of tiles requested per zoom in MUSeGen and HELP/Custom HELP, respectively. As in the pan, the number of tiles requested in MUSeGen varies significantly more, and this is closely related to the distribution of screen resolutions.

In summary, these results demonstrate that user behavior patterns generated by MUSeGen and HELP are very different, even when HELP is parameterized with the same empirical data used to parameterized MUSeGen. On average, the number of issued operations per session in MUSeGen is four times higher than that in HELP. The size of the pan movements in HELP is fixed, and the pans are limited to four directions. In addition, MUSeGen sessions request, on average, twice more tiles than HELP sessions. Since the behavior patterns generated by MUSeGen are richer and match closely with those observed in actual data (as shown in Subsection VI-A), there is strong evidence that HELP can not generate user behavior with the same realism as MUSeGen. Thus, we conclude that MUSeGen is a valuable tool to help system administrators analyze performance and scalability issues related to WMSs.

VII. CASE STUDY: PERFORMANCE EVALUATION OF A REAL WMS UNDER DIFFERENT WORKLOADS

This section shows how the differences between HELP and MUSeGen affect performance evaluation in practice. To achieve this goal, we present a performance evaluation of a real WMS under workloads generated by (i) a stress testing software (RTRM model); (ii) HELP; and (iii) MUSeGen.



(a) MUSeGen (b) HELP/Custom HELP

FIGURE 15. Comparison of MUSeGen, HELP, and Custom HELP in relation to the number of tiles requested per zoom.

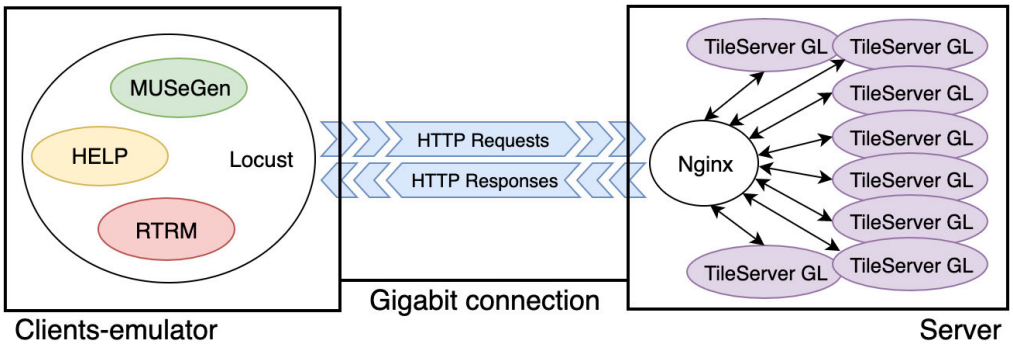


FIGURE 16. Testbed used in the performance evaluation study.

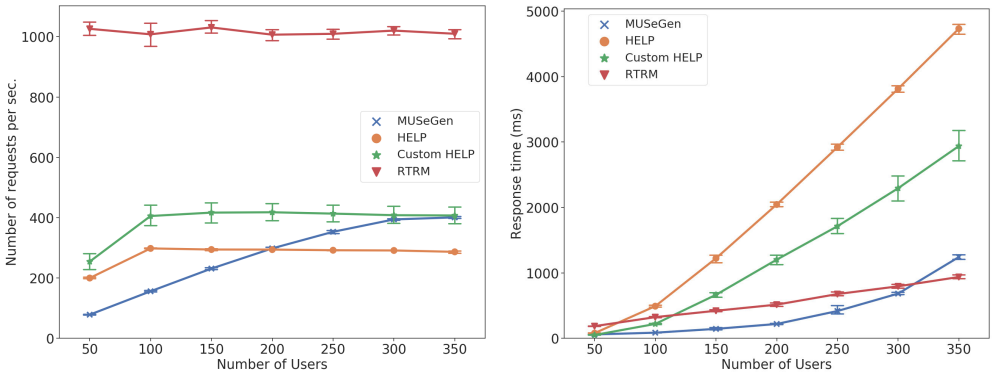
To perform this evaluation, we employ the testbed illustrated in Fig. 16. The testbed consists of two computers connected through a Gigabit connection. The server runs one instance of the Nginx Web server,[5] which operates as a load balancer for eight instances of the TileServer GL,[6] an open-source and widely used map server for vector and raster maps. The tiles are effectively served by these instances. The clients-emulator executes the workload generators, i.e., MUSeGen, HELP, Custom HELP, and RTRM, one at a time. All the workload generators were implemented inside Locust,[7] a load and performance testing framework written in Python. The tile server instances handle the requests from and return the responses to the Web server. The latter returns the responses to the clients. Table 8 depicts the hardware configuration and software versions used in our case study.

We adopt the following methodology to conduct this evaluation. To mimic the behavior of a conventional Web browser, each client has a local cache. Each client can also establish up to six TCP/HTTP connections simultaneously. Each experiment consists of generating a workload for 10 minutes with a fixed number of users. For HELP, Custom HELP, and MUSeGen, a new user session is instantiated as soon as a previous one is finished. The number of users varies from 50 up to 350, increasing by 50 for each new experiment. Each experiment is executed 30 times, and the average of the following metrics are collected at the server-side: the number

TABLE 8. Hardware and Software Employed in the Testbed.

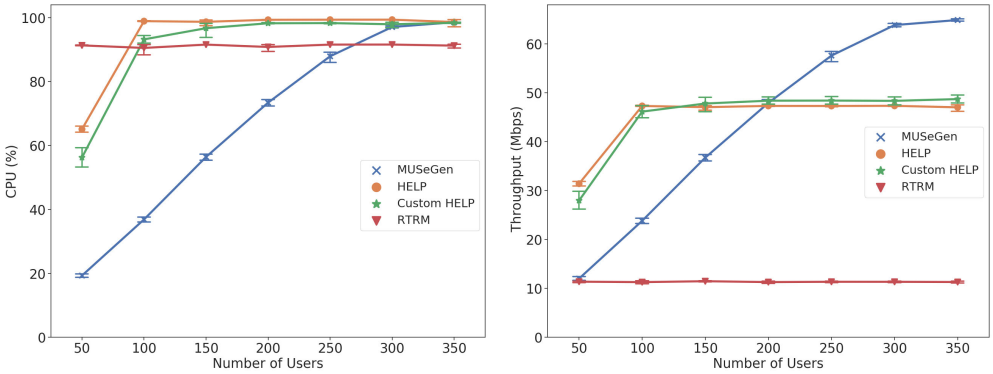
	Clients-emulator	Server
Hardware	Intel i5-6400 @ 2.7 GHz, 4 cores and 4 threads 48 GB of RAM, DDR4 2400 MHz 240 GB of Solid-state drive, 530 MB/s R, 440 MB/s W	AMD Ryzen 7 2900X @ 3.7 GHz, 8 cores and 8 threads 64 GB of RAM, DDR4 2400 MHz 512 GB of Solid-state drive, 3500 MB/s R, 2300 MB/s W
Software	Ubuntu Linux 16.04 LTS kernel 4.4.0-166-generic Locust 0.13.2	Ubuntu Linux 18.04 LTS kernel 5.0.0-32-generic Nginx 1.14.0 TileServer GL 2.5.0

[5]https://www.nginx.com
[6]http://tileserver.org
[7]https://locust.io



(a) Number of requests per second (b) Response time (in milliseconds)

FIGURE 17. Comparison of MUSeGen, HELP, Custom HELP, and RTRM in relation to: (a) the average number of requests per second generated by each tool; and (b) the average server response time (in milliseconds) in the server-side.



(a) CPU usage (in percentage form) (b) Network throughput (in Mbps)

FIGURE 18. Comparison of workload generators in relation to (a) CPU utilization; and (b) network throughput.

of requests per second, response time, network throughput, and CPU usage. These average values are presented with confidence interval bounds at a confidence level of 95%.

Fig. 17(a) presents the average number of requests per second (arriving in the server) produced by each tool, while Fig. 17(b) shows the average server response time (in milliseconds). HELP and Custom HELP have similar trends, but different absolute values, for both metrics. This difference is due to the session lengths and the number of issued pans, which are larger in Custom HELP. Thus, Custom HELP generates more requests per second than HELP. Indeed, HELP and Custom HELP achieves the maximum number of requests per second supported by the server with 100 users. This is supported by Fig. 17(b), which shows that the average server response time for HELP and Custom HELP increases rapidly above 100 users. RTRM achieves the maximum number of requests per second with 50 users, but most of the requests are for tiles with little information. Thus, these requests are processed quickly by the server. As a consequence, the response time under RTRM increases slowly in comparison with the other workload generators. MUSeGen, on the other hand, presents the most consistent behavior in which the number of requests, and the server response time, starts small and increases as the number of users grows.

Figs. 18(a) and 18(b) show, respectively, the average CPU usage in the server, and the average network throughput (in Mbps) from the server to the clients-emulator during the experiments. Both metrics are strongly influenced by the number of requests (per second) and by the density of the browsed areas. RTRM keeps the CPU utilization high (above 90%) but constant since its tile distribution is homogeneous. Due to this homogeneity, the network throughput is constant. Also, since most of the RTRM requests are for tiles with little information, RTRM leads to the smallest network throughput. These results highlight the limitations of stress testing tools to evaluate the performance of WMSs.

On the other hand, MUSeGen, HELP, and Custom HELP present similar trends, but with different absolute values, for CPU utilization and average network throughput. In HELP and Custom HELP, the saturation of the system occurs with approximately 100 concurrent users. At this point, CPU utilization is nearly 100% and an increase in the number of users does not result in the growth of the number of requests (Fig. 17(a)) or in the increase of the network throughput (Fig. 18(b)). In MUSeGen, the saturation of the system occurs with approximately 300 concurrent users. These results show that system capacity under MUSeGen workload is almost three times higher than that observed

under HELP and Custom HELP. This happens because the number of tiles requested per zoom or pan in MUSeGen is higher than in HELP/Custom HELP. This, in turn, increases the browse's cache hit, and most of the requests are satisfied by the browser's cache. As a consequence, the tile server can serve more users concurrently. These results confirm the value of MUSeGen in helping system administrators analyze the performance of WMSs.

VIII. CONCLUSION

This article used data collected anonymously from sessions of a client application of Google Maps to devise a model that describes how users of desktop terminals navigate in a Web map. Based on this model, we implemented a workload generator called MUSeGen. MUSeGen is the first tool in the literature that simulates WMS user behavior based on data gathered from real WMS sessions. We validated our workload generator against real traces and compared it with HELP. We showed that the MUSeGen workload is in close agreement with real traces. We also showed that our tool generates more realistic workloads than HELP. Finally, we illustrated how the workload patterns produced by a stress testing tool, HELP, and MUSeGen impact the performance evaluation of a real WMS in practice. We offer MUSeGen to the community as free software, as a valuable tool to simulate the behavior of WMS users.

We highlight that the set of operations available in a WMS is broad, but the operations implemented in MUSeGen are the ones most used in a navigation session. As future work, we intend to characterize the workload patterns of mobile terminals. This work has considered user behavior patterns focusing on website visitor users. Another interesting work is to introduce other types of WMS users (e.g., content authors users) and analyze user behavior patterns from other perspectives than that of the workload. Process Mining techniques [29] can help in this context.

REFERENCES

[1] Google. (2020). *Google Maps Platform*. Accessed: Sep. 20, 2020. [Online]. Available: https://cloud.google.com/maps-platform/

[2] F. Malandrino, C.-F. Chiasserini, G. Avino, M. Malinverno, and S. Kirkpatrick, "From megabits to CPU ticks: Enriching a demand trace in the age of MEC," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 43–50, Mar. 2020.

[3] M. Curiel and A. Pont, "Workload generators for Web-based systems: Characteristics, current status, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1526–1546, 2nd Quart., 2018.

[4] R. F. Tomlinson, *Thinking About GIS: Geographic Information System Planning for Managers*, vol. 1. Redlands, CA, USA: ESRI, 2007.

[5] X. Guan, B. Cheng, A. Song, and H. Wu, "Modeling users' behavior for testing the performance of a Web map tile service," *Trans. GIS*, vol. 18, pp. 109–125, Nov. 2014.

[6] V. G. Braga, S. L. Corr, V. J. D. S. Rodrigues, and K. V. Cardoso, "Characterizing user behavior on Web mapping systems using real-world data," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2018, pp. 01056–01061.

[7] Google. *Google Maps Metrics and Infographics*. Accessed: Feb. 20, 2021. [Online]. Available: https://sites.google.com/a/pressatgoogle.com/google-maps-for-iphone/google-maps-metrics

[8] S. Li, S. Dragicevic, and B. Veenendaal, *Advances in Web-Based GIS, Mapping Services and Applications*. Boca Raton, FL, USA: CRC Press, 2011.

[9] W3C. (2010). *Use Cases and Requirements for Standardizing Web Maps*. [Online]. Available: https://maps4html.org/HTML-Map-Element-UseCases-Requirements/#use-cases

[10] J. T. Sample and E. Ioup, *Tile-Based Geospatial Information Systems:Principles and Practices*. New York, NY, USA: Springer, 2010.

[11] J. Snyder, *Map Projections—A Working Manual*. Washington, DC, USA: U.S. Government Printing Office, 1994.

[12] P. Barford and M. Crovella, "Generating representative Web workloads for network and server performance evaluation," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 26, no. 1, pp. 151–160, Jun. 1998.

[13] D. Krishnamurthy, J. A. Rolia, and S. Majumdar, "A synthetic workload generation technique for stress testing session-based systems," *IEEE Trans. Softw. Eng.*, vol. 32, no. 11, pp. 868–882, Nov. 2006.

[14] M. C. Calzarossa, L. Massari, and D. Tessera, "Workload characterization: A survey revisited," *ACM Comput. Surveys*, vol. 48, no. 3, pp. 1–43, Feb. 2016.

[15] K. Goševa-Popstojanova, A. D. Singh, S. Mazimdar, and F. Li, "Empirical characterization of session–based workload and reliability for Web servers," *Empirical Softw. Eng.*, vol. 11, no. 1, pp. 71–117, 2006.

[16] (2018). *MAPLARGE*. Accessed: Jan. 24, 2020. [Online]. Available: https://maplarge.com/

[17] B. Romoser, R. Fares, P. Janovics, X. Ruan, X. Qin, and Z. Zong, "Global workload characterization of a large scale satellite image distribution system," in *Proc. 31st Int. Perform. Comput. Commun. Conf.*, 2012, pp. 368–375.

[18] S. Quinn and M. Gahegan, "A predictive model for frequently viewed tiles in a Web map," *Trans. GIS*, vol. 14, no. 2, pp. 193–216, Apr. 2010.

[19] S. Yeşilmurat and V. İşler, "Retrospective adaptive prefetching for interactive Web GIS applications," *GeoInformatica*, vol. 16, no. 3, pp. 435–466, Jul. 2012.

[20] M. R. Vieira, P. Bakalov, E. Hoel, and V. J. Tsotras, "A spatial caching framework for map operations in geographical information systems," in *Proc. IEEE 13th Int. Conf. Mobile Data Manage.*, Jul. 2012, pp. 89–98.

[21] S. Pan, Y. Chong, H. Zhang, and X. Tan, "A global user-driven model for tile prefetching in Web geographical information systems," *PLoS ONE*, vol. 12, pp. 1–22, Jan. 2017.

[22] V. G. Braga, W. B. de Oliveira, V. J. do Sacramento, and K. V. Cardoso, "Understanding and modeling the behavior of Web map users," *J. Inf. Data Manage.*, vol. 6, no. 1, pp. 92–103, 2015.

[23] R. B. D'Agostino and M. A. Stephens, Eds., *Goodness-of-Fit Techniques*. New York, NY, USA: Marcel Dekker, 1986.

[24] R. Jain, *Book Review: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Hoboken, NJ, USA: Wiley, 1991.

[25] W. N. Venables and B. D. Ripley, *Modern Applied Statistics With S*. New York, NY, USA: Springer, 2010.

[26] D. G. Feitelson, "Workload modeling for performance evaluation," in *Performance Evaluation of Complex Systems: Techniques and Tools*. Berlin, Germany: Springer, 2002, pp. 114–141.

[27] StatCounter Global Stats. (2015). *Top 10 Desktop Screen Resolutions From July to Aug 2015*. Accessed: Aug. 25, 2015. [Online]. Available: http://gs.statcounter.com/desktop-resolution-ww-monthly-201507-201508-b-ar

[28] R. García, J. P. de Castro, E. Verdú, M. J. Verdú, and L. M. Regueras, "Web map tile services for spatial data infrastructures: Management and optimization," in *Cartography—A Tool for Spatial Analysis*. Rijeka, Croatia: InTech, 2012, pp. 25–35.

[29] W. van der Aalst, *Process Mining: Data Science in Action*. Berlin, Germany: Springer, 2016.

VINÍCIUS GONÇALVES BRAGA received the B.S. degree in software engineering and the M.Sc. degree in computer science from the Universidade Federal de Goiás (UFG), in 2013 and 2015, respectively.

SAND LUZ CORREA received the degree in computer science from the Universidade Federal de Goiás, in 1994, the M.Sc. degree in computer sciences from the University of Campinas (UNICAMP), in 1997, and the Ph.D. degree in informatics from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), in 2011. In 2015, she spent her sabbatical at Virginia Tech, USA and at Inria Saclay Research Centre, France, in 2020. She has been a Professor and a Researcher with the Institute of Informatics, Universidade Federal de Goiás (UFG), since 2010, where she is currently an Associate Professor. She has participated in some international research projects (including two from joint calls BR-EU). Her research interests include cloud computing, SDN, data plane programmability, sensor systems, and smart city platforms.

KLEBER VIEIRA CARDOSO received the degree in computer science from the Universidade Federal de Goiás, in 1997, and the M.Sc. and Ph.D. degrees in electrical engineering from COPPE, Universidade Federal do Rio de Janeiro, in 2002 and 2009, respectively. In 2015, he spent his sabbatical at Virginia Tech, USA, and at Inria Saclay Research Centre, France, in 2020. He has been a Professor and a Researcher with the Institute of Informatics, Universidade Federal de Goiás (UFG), since 2009, where he is currently an Associate Professor. He has participated in some international research projects (including two from joint calls BR-EU) and coordinated several national-sponsored research and development projects. His research interests include wireless networks, software-defined networks, virtualization, resource allocation, and performance evaluation.

ALINE CARNEIRO VIANA received the Habilitation degree and the Ph.D. degree in computer science from UPMC, Sorbonne Universités, France, in 2005 and 2011, respectively. She was with the TKN Group, TU-Berlin, Germany after a one-year sabbatical leave. She is currently a permanent Research Director (DR) with Inria, where she also leads the TRiBE Team. Her research interests include the design of solutions for tactful networking, smart cities, and mobile and self-organizing networks with the focus on human behavior analysis. She was a recipient of the French Scientific Excellence Award since 2015, and for six years now and was nominated in 2016 as one of the "ten women in networking/communications that you should watch" (1st-year nomination of N2Women community).